UTF-8 Validation (View)

Given an integer array `data` representing the data, return whether it is a valid **UTF-8** encoding (i.e. it translates to a sequence of valid UTF-8 encoded characters).

A character in **UTF8** can be from **1 to 4 bytes** long, subjected to the following rules:

1. For a **1-byte** character, the first bit is a `0`, followed by its Unicode code.
2. For an **n-bytes** character, the first `n` bits are all one's, the `n + 1` bit is `0`, followed by `n - 1` bytes with the most significant `2` bits being `10`.

This is how the UTF-8 encoding would work:

```
   Number of Bytes   |        UTF-8 Octet Sequence

                     |              (binary)

   --------------------+-----------------------------------------

          1          |   0xxxxxxx

          2          |   110xxxxx 10xxxxxx

          3          |   1110xxxx 10xxxxxx 10xxxxxx

          4          |   11110xxx 10xxxxxx 10xxxxxx 10xxxxxx
```

`x` denotes a bit in the binary form of a byte that may be either `0` or `1`.

Note: The input is an array of integers. Only the **least significant 8 bits** of each integer is used to store the data. This means each integer represents only 1 byte of data.

Example 1:

```
Input: data = [197,130,1]

Output: true

Explanation: data represents the octet sequence: 11000101 10000010 00000001.

It is a valid utf-8 encoding for a 2-bytes character followed by a 1-byte
character.
```

Example 2:

```
Input: data = [235,140,4]

Output: false

Explanation: data represented the octet sequence: 11101011 10001100 00000100.
```

```
The first 3 bits are all one's and the 4th bit is 0 means it is a 3-bytes
character.

The next byte is a continuation byte which starts with 10 and that's correct.

But the second continuation byte does not start with 10, so it is invalid.
```

Constraints:

- `1 <= data.length <= 2 * 10`^{4}
- `0 <= data[i] <= 255`